Date=03/08/2020
Lecture By=Shubham Joshi
Subject ⇒ Oops Principles

IN PREVIOUS LECTURE **(QUICK RECAP) Date-31/07/2020**	**In Today's Lecture (Overview)**
Oops in python Class In python Questions for self practice	Principle Of Oops 1.Polymorphism 2.Encapsulation 3.Inheritance Super Function In Python Oops 4.Abstraction Questions For Self Practice // CC for the day

In today's lecture we learned about principles of oops that are explained below

Principle Of Oops

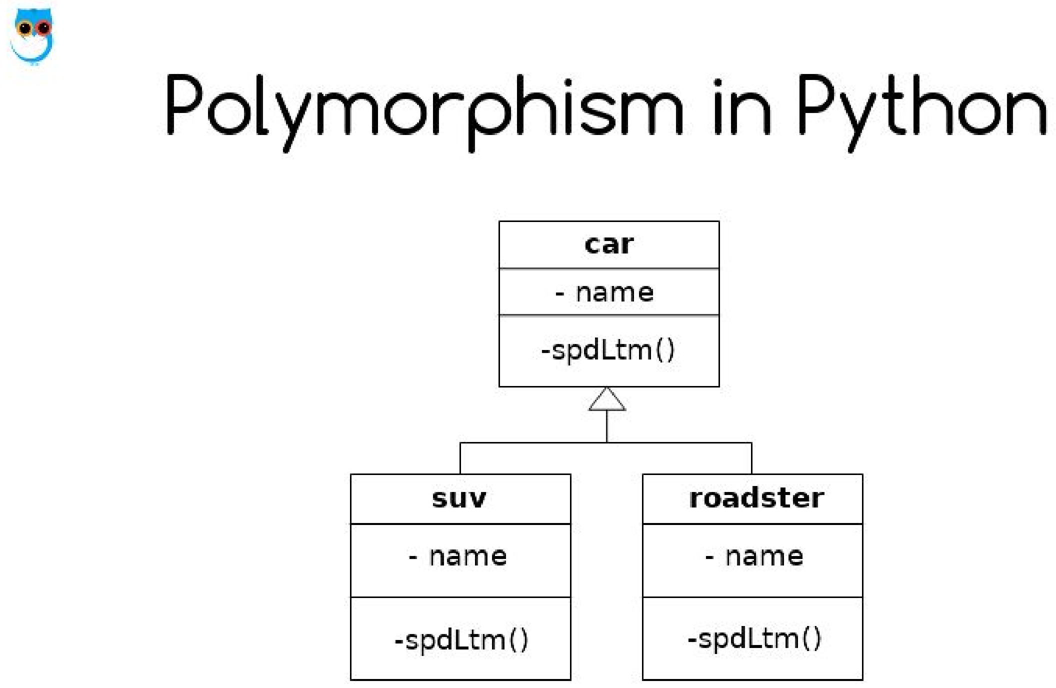
1.Polymorphism

Polymorphism in python defines methods in the child class that have the same name as the methods in the parent class

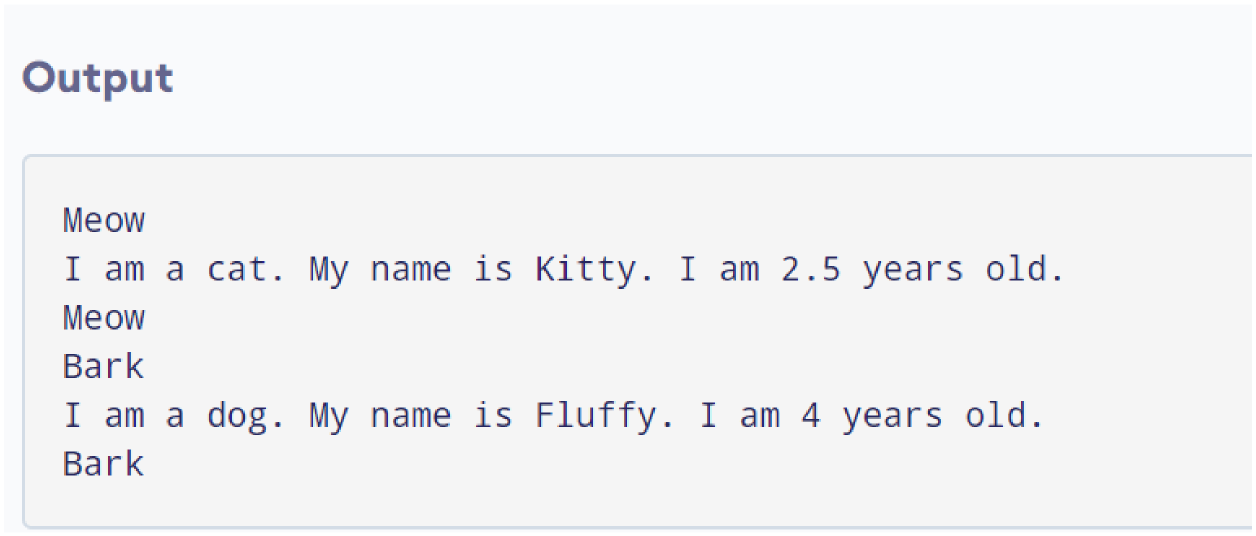
The word polymorphism means having many forms.

In short

=polymorphism means same function name (but different signatures) being uses for different types.



```python
class Cat:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def info(self):
        print(f"I am a cat. My name is {self.name}. I am {self.age} years old.")

    def make_sound(self):
        print("Meow")


class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def info(self):
        print(f"I am a dog. My name is {self.name}. I am {self.age} years old.")

    def make_sound(self):
        print("Bark")


cat1 = Cat("Kitty", 2.5)
dog1 = Dog("Fluffy", 4)

for animal in (cat1, dog1):
    animal.make_sound()
    animal.info()
    animal.make_sound()
```



Output

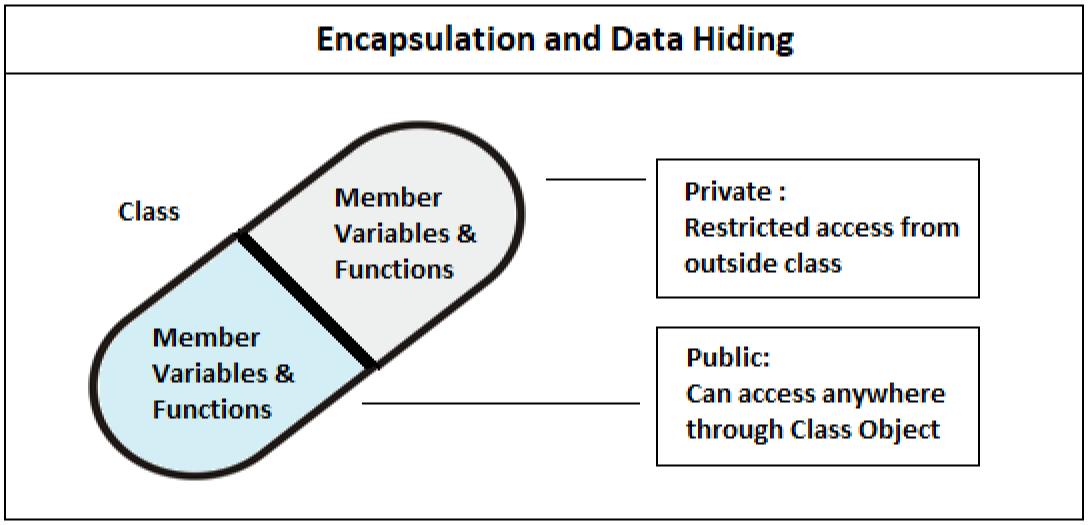
```
Meow
I am a cat. My name is Kitty. I am 2.5 years old.
Meow
Bark
I am a dog. My name is Fluffy. I am 4 years old.
Bark
```

“Click Here” to know more about it

2.Encapsulation

Other programming languages have protected class methods too, but Python does not. Encapsulation gives you more control over the degree of coupling in your code, it allows a class to change its implementation without affecting other parts of the code.

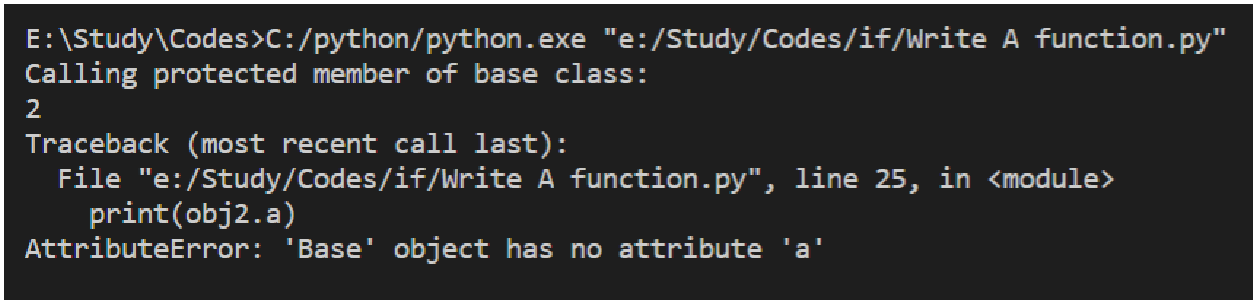
To make it private or hidden you have to Put Double underscore like this ” __ “


Encapsulation and Data Hiding
Class
Member Variables & Functions
Member Variables & Functions
Private : Restricted access from outside class
Public: Can access anywhere through Class Object

```
# Creating a base class
class Base:
    def __init__(self):

        # Protected member
        self._a = 2

# Creating a derived class
class Derived(Base):
    def __init__(self):

        # Calling constructor of
        # Base class
        Base.__init__(self)
        print("Calling protected member of base class: ")
        print(self._a)

obj1 = Derived()

obj2 = Base()

# Calling protected member
# Outside class will  result in
# AttributeError
```

```
print(obj2.a)
```

Output



```
E:\Study\Codes>C:/python/python.exe "e:/Study/Codes/if/Write A function.py"
Calling protected member of base class:
2
Traceback (most recent call last):
  File "e:/Study/Codes/if/Write A function.py", line 25, in <module>
    print(obj2.a)
AttributeError: 'Base' object has no attribute 'a'
```

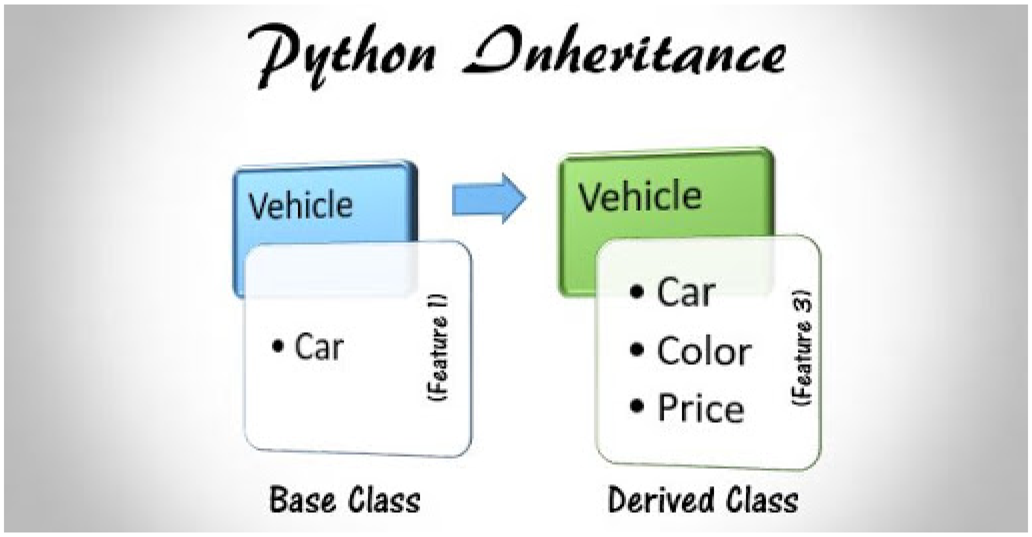
“Click here” to Know more about it

3.Inheritance

Inheritance allows us to define a class that inherits(Contains) all the methods and properties from another class.

Parent class== is the class being inherited from, also called base class.

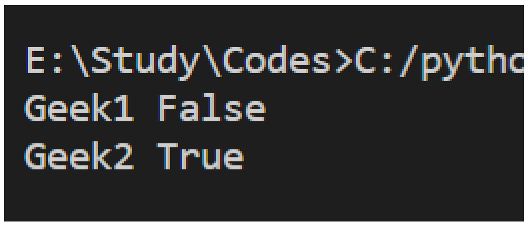
Child class== is the class that inherits from another class, also called derived class.



```
class Person(object):


    # Constructor
```

```python
    def __init__(self, name):
        self.name = name

    # To get name
    def getName(self):
        return self.name

    # To check if this person is employee
    def isEmployee(self):
        return False


# Inherited or Sub class (Note Person in bracket)
class Employee(Person):

    # Here we return true
    def isEmployee(self):
        return True

# Driver code
emp = Person("Geek1")  # An Object of Person
print(emp.getName(), emp.isEmployee())

emp = Employee("Geek2") # An Object of Employee
print(emp.getName(), emp.isEmployee())
```

Output



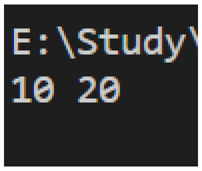
```
E:\Study\Codes>C:/pytho
Geek1 False
Geek2 True
```

Super Function In Python Oops

We can also access parent class members using super.

```
class Base(object):

    # Constructor
    def __init__(self, x):
        self.x = x

class Derived(Base):

    # Constructor
    def __init__(self, x, y):

        ''' In Python 3.x, "super().__init__(name)"
            also works'''
        super(Derived, self).__init__(x)
        self.y = y

    def printXY(self):

       # Note that Base.x won't work here
       # because super() is used in constructor
       print(self.x, self.y)


# Driver Code
d = Derived(10, 20)
d.printXY()
```

Output



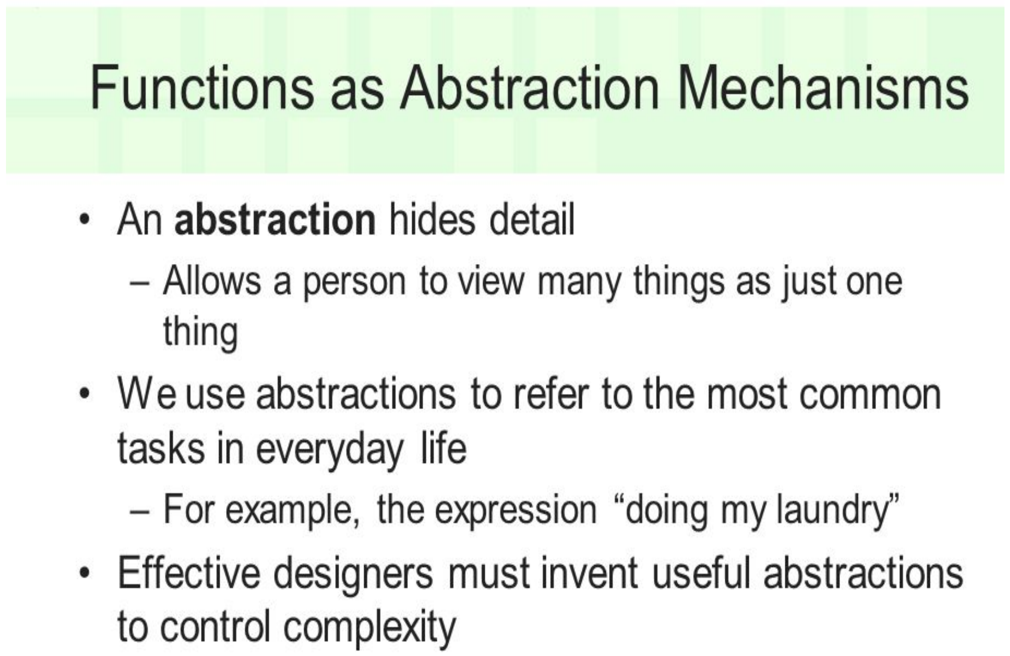
```
E:\Study\
10 20
```

[“Click Here”](#)To know more about It

4.Abstraction

Abstraction in Python is the process of hiding the real implementation of an application from the user and emphasizing only on usage of it
For example, consider you have bought a new electronic gadget. Along with the gadget, you get a user guide, instructing how to use the application, but this user guide has no info regarding the internal working of the gadget.

In short
=It hides The complexity And only shows the main Features



Functions as Abstraction Mechanisms

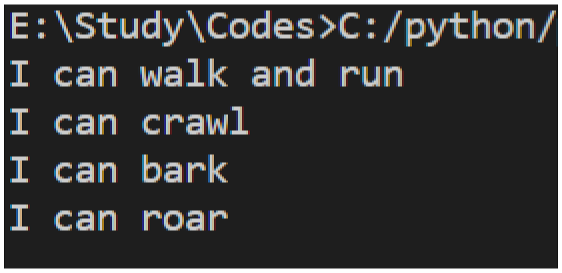
- An **abstraction** hides detail
  - Allows a person to view many things as just one thing
- We use abstractions to refer to the most common tasks in everyday life
  - For example, the expression "doing my laundry"
- Effective designers must invent useful abstractions to control complexity

```
from abc import ABC, abstractmethod
class Animal(ABC):

    def move(self):
        pass

class Human(Animal):

    def move(self):
        print("I can walk and run")
```

“Click Here” For More Example

```
class Snake(Animal):

    def move(self):
        print("I can crawl")

class Dog(Animal):

    def move(self):
        print("I can bark")

class Lion(Animal):

    def move(self):
        print("I can roar")

# Driver code
R = Human()
R.move()

K = Snake()
K.move()

R = Dog()
R.move()

K = Lion()
K.move()
```

Output



```
E:\Study\Codes>C:/python/
I can walk and run
I can crawl
I can bark
I can roar
```

“Click Here” For More Example

Questions For Self Practice // CC for the day

Q1. Write down about the oops principles ?
Q2. Implement a real life bank by using oops concepts. Eg create classes as Bank, Depositor, Clerk, BankAccount